VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2009 David Říhošek

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Simulátory počítačových sítí
Simulators of Computer Networks

2009 David Říhošek

Zadání

// jestli ty úvodní stránky mají být s rámečkem nebo bez?
potom velikost písma na těch stánkách o zadání se píše o základní velikosti 11 ale o nadpisech nic tak jsem je zvolil přiměřeně, nevím jestli to tak může být

Prohlášení studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Přerově dne 7.5 2009 David Říhošek

Abstrakt

Aby se pro testování síťových aplikací a protokolů nemusely používat fyzické síťové komponenty, byly vyvinuty síťové simulátory a emulátory. S jejich pomocí jsme schopni otestovat a simulovat chování těchto protokolů, aplikací a zařízení na jakýchkoliv sítích, které je simulátor schopen vytvořit. Bez nutnosti velkých investic jsme tedy schopni provádět testy těchto aplikací a ladit jejich případné nedostatky. Síťové simulátory slouží hlavně k simulaci chování sítí a síťových protokolů s určitými vlastnostmi. Síťové emulátory jsou zase vhodnější pro testování chování jednotlivých síťových aplikací ve specifických podmínkách, které je schopen síťový emulátor vytvořit. Tato práce popisuje jak síťové simulátory, tak síťové emulátory a poskytuje k nim jednoduché vysvětlení jejich fungování a jejich srovnání.

Klíčová slova

Síťový simulátor, Síťový emulátor, simulace, emulace, Ns-1, Ns-2, Ns-3, Net:Netem, WANem, WANulator.

Abstract

The network simulators and the network emulators have been worked out to avoid using the physical network components by testing of network applications and protocols. By means of them, we are able to test and to simulate behaviour of these protocols, applications and devices on any network, that can be created by the simulator. Without necessity of large investments, we are able to test these applications and to tune their possible deficiencies. The network simulators can be used principally to simulation of network behaviour and network protocols having certain properties. The network emulators are more fit for testing of behaviour of the respective network applications in specific conditions which can be created by the network emulator. This dissertation describes both the network simulators and the network emulators and gives an easy explanation to their function and their comparison.

Keywords

Network simulator, Network emulator, simulation, emulation, Ns-1, Ns-2, Ns-3, Net:Netem, WANem, WANulator.

Seznam použitých symbolů a zkratek

CBQ	-	Class-Based Queueing, druh fronty
CBR	-	Constant Bitrate
CSMA/CD	-	Carrier sense multiple access with collision detection
DRR	-	Druh fronty
FIFO	-	First In First Out, druh fronty
IPv4	-	IP protokol verze 4
IPv6	-	IP protokol verze 6
LAN	-	Local Area Network
RED	-	Random Early Detection, druh fronty
RTP	-	Real-time Transport Protocol
RTCP	-	Real-time Transport Protocol
SRM	-	Storage Resource Manager
SFQ	-	Stochastic Fairness Queuing, druh fronty
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol
VoIP	-	Voice over Internet Protocol
Wi-Fi	-	Wireless Fidelity, Wireless LAN
WiMAX	-	Worldwide Interoperability for Microwave Access
WRR	-	Druh fronty

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Tato bakalářská práce se věnuje problematice simulování počítačových sítí na lokálním stroji, bez nutnosti používat fyzické síťové komponenty. Bakalářská práce je rozdělena na dvě hlavní části. První se zabývá síťovými simulátory, druhá potom síťovými emulátory.

Na začátku první části je uveden úvod do simulace počítačových sítí, po něm následují kapitoly o jednotlivých síťových simulátorech: Ns-1, Ns-2 a Ns-3. Každým z nich se zabývá samostatná kapitola. Je zde popsána instalace a základy obsluhy těchto programů. V příloze jsou uvedeny okomentované skripty sloužící pro sestavování simulací. Na závěr je uvedeno srovnání všech tří simulátorů, jejich výhody a nevýhody.

V další části bakalářské práce je probírána problematika síťových emulátorů. Na úvod je vysvětlena problematika síťového emulátoru. Dále následují kapitoly o třech různých síťových emulátorech: Net:Netem, WANem a Wanulator. Tak jako u síťových simulátorů tak, i zde je každému emulátoru věnována jedna kapitola, kde je popsáno jeho fungování, instalace a používání. Na závěr je opět uvedeno srovnání všech tří emulátorů.

2. Síťový Simulátor

V oblasti komunikací a počítačových sítí je simulace sítí technika, při které se pomocí simulačních programů testují zadané topologie a uvnitř nich vzájemné vztahy mezi různými subjekty např. koncová zařízení, směrovače, přenášené pakety, datové linky, atd. Pomocí matematických modelů a skutečných zkušeností z existujících sítí se potom sestavuje simulace chování sítě se zadanými parametry. Simulovat se dají jak klasické drátové sítě, tak i bezdrátové a satelitní sítě. Většina simulátorů pracuje s nejpoužívanějšími standardy, jako jsou IPv4, IPv6, UDP a TCP. Sítové simulátory jsou dostupné buď zdarma, anebo komerčně. V této práci se budu zabývat simulátory, které jsou zdarma a volně dostupné. Tyto simulátory běží na Linuxu a většinou se ovládají pomocí skriptů, do kterých se zapíše, co se má při simulaci vykonávat. Komerční simulátory běží jak na platformě Linux tak i Windows a obsahují grafické rozhraní.

2.0.1. Technika simulace

Většina síťových simulátorů využívá techniku diskrétních událostí, v níž je uložen seznam nevyřízených událostí. Tyto události jsou spouštěny v předem zadaném pořadí, přičemž předchozí událost po svém provedení spouští událost následující.

2.1. Simulátor Ns-2 (Network Simulator 2)

2.1.1. Úvod

Síťový simulátor Ns-2 je v současné době nejpoužívanější simulátor z oblasti volně šiřitelných simulátorů a Linuxu, je možné jej zdarma stáhnout a volně používat. Využívá se k simulaci počítačových sítí. Dají se zde simulovat jak pevné drátové sítě, tak bezdrátové sítě typu Wi-Fi a dokonce i sítě využívající komunikaci pomocí satelitů. Ns-2 byl napsán v C++ a pro zadávání simulací využívá skriptů jazyka TCL. Uživatel nejdříve popíše simulaci pomocí OTCL skriptu, ten potom spustí v hlavním programu Ns-2, a ten skript vykoná. Součástí tohoto simulátoru jsou také nástroje Nam a Xgraph. Nam se využívá k vizualizaci výsledných simulací, které vytvořil program Ns-2 z TCL skriptů, pomocí Xgraph může uživatel zobrazit například grafy, které ukazují vytížení jednotlivých zásobníků po cestě sítí.

2.1.2. Podporované prvky

Protokoly: IP,TCP,RTP/RTCP,UDP, SRM, 802.11 MAC, 802.3 MAC

Směrování: Statický unicast, dynamický unicast(distance - vector) a multicast

Topologie: Uzly a spoje

Chybovost: Generování chyb na linkách od 0 až do 100% po kroku 1%

Tvorba front a plánování: SFQ, RED, FIFO/drop-tail, CBQ, WRR, DRR

Bezdrátové sítě: Ad-Hoc směrování a Mobile IP: AODV
Sensor-Mac, WiMAX
Kontrola spotřeby v mobilních sítích

2.1.3. Stažení a instalace Ns-2

Při stahování a sestavování simulátoru jsou dvě možnosti. Buď se dá stáhnout a instalovat jako jeden velký instalační balíček 'all-in-one', nebo po jednotlivých částech a tyto části potom sestavovat samostatně. Já jsem v této práci využil balíčku 'all-in-one'. Balíček 'all-in-one' je sice náročnější na použité místo, instalují se všechny komponenty, i ty které nikdy nevyužijete ale je rychlejší pro instalaci a méně náročný na čas a schopnosti uživatele. Pokud byste se rozhodli pro instalaci po částech, je třeba stáhnout všechny instalační balíčky, nainstalovat je jednotlivě a potom je propojit.

Instalační balíčky je možno stáhnout z: http://www.isi.edu/nsnam/ns/ns-build.html

Pokud by při instalaci nastali jakékoliv potíže, je možno se podívat na:
http://www.isi.edu/nsnam/ns/ns-problems.html

2.1.3.1. Postup instalace

Pokud jste si stáhli balíček 'all-in-one', potom je postup instalace jednoduchý. Stačí mít nainstalován kompilátor pro jazyk C (např. GCC), rozbalit stažený archiv do jakékoliv složky, v této složce si spustit okno terminálu a tam zadat příkaz *./install* Program potom sám rozbalí a propojí veškeré instalační zdroje. Jakmile bude instalace dokončena, na obrazovce se zobrazí informace co ještě potřeba udělat. Do proměnného prostředí LD_LIBRARY_PATH se musí vložit složka bin a lib. Provádí se to pomocí příkazu:

`setenv LD_LIBRARY_PATH <cesta>` pokud používáte csh,

anebo pomocí příkazu:

`export LD_LIBRARY_PATH=<cesta>` pokud používáte bash.

Dále je potom potřeba zapsat cestu k TCL knihovně do proměnné TCL_LIBRARY, aby nevznikaly problémy při spouštění programů Ns-2 a Nam.

Jakmile budeme mít tohle hotovo, můžeme spustit testy. Přepneme se do složky ns-2.33, tam spustíme testy pomocí *./validate* . Jakmile proběhnou všechny testy, je instalace dokončena a program připraven na používání.

2.1.4. První kroky

2.1.4.1. Spuštění NS

Program Ns-2, který vykonává zadané TCL skripty, se spouští pomocí terminálu. Terminál je třeba spustit v adresáři, kde máme nainstalován program Ns-2, nebo se do něj přepnout. (*.../složka instalace /ns-2.33*) anebo jeho zástupce (*.../složka instalace/bin*). V terminálu je třeba zadat: *./ns ´název skriptu.tcl´* pokud máme skript umístěn ve stejné složce jako program, nebo: *./ns ´cesta ke skriptu/název skriptu.tcl´* potom už všechno závisí na zadaném TCL skriptu, podle kterého program vykonává operace. Výstupem programu může být výpis na obrazovku terminálu, výstupní soubor pro nástroje Nam nebo Xgraph anebo všechno dohromady.

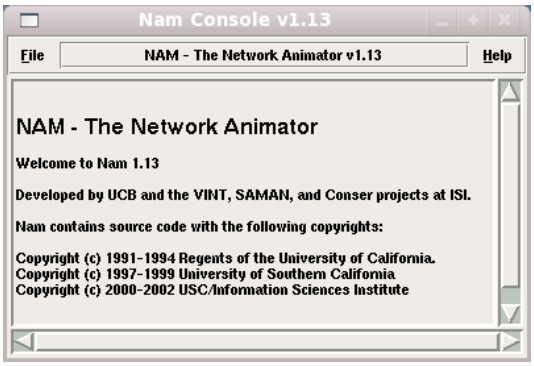
2.1.4.2. Spuštění Nam

Program Nam, který se používá k vizualizaci výstupních dat z programu Ns-2, se dá zapnout dvěma způsoby. Buď ručně přes grafické rozhraní z cesty *.../složka instalace/bin/nam* a nebo přímo z terminálu, podobně jako program Ns-2. Z terminálu se program Nam zapíná pomocí příkazu *./nam´nazev_animace.nam´* pokud chceme otevřít přímo nějakou animaci, anebo jednoduše pouze *./nam* pouze pro otevření aplikace bez vstupních souborů .

2.1.4.3. Spouštění Xgraph

Program Xgraph, který slouží k animaci různých grafů vytvořených programem Ns-2, se spouští v terminálu ve složce *.../složka instalace/bin* pomocí příkazu *./xgraph* a jako parametr se zadává cesta a název výstupního souboru z programu Ns-2, pro který chceme vytvořit graf.

Obr. 1: Hlaví okno programu Nam



2.1.5. Tvorba skriptu pro Ns-2

Pro tvorbu skriptů simulujících skutečné síťové prostředí se používá skriptovací jazyk TCL. Tvorbu skriptu ukážu na příkladě, ve kterém budou pouze dvě koncová zařízení - uzly a ty si budou mezi sebou posílat data. Tento skript je možné psát v jakémkoliv textovém editoru.

2.1.5.1. Vzorový soubor

Jako první krok je třeba si vytvořit objekt simulace. To se dělá pomocí příkazu:

```
set ns [Nova Simulace]
```

Dále je třeba si vytvořit výstupní soubor, do kterého se budou logovat data, která simulátor vypočítá. Tento soubor se potom využije v programu NAM pro vizualizaci simulace. Pomocí prvního řádku tvoříme soubor a pomocí druhého do něj logujeme vše co se vypočítá.

```
set nf [open Vystupni_Soubor.nam w]
$ns namtrace-all $nf
```

Dále potom vytvoříme koncovou proceduru, která bude ukončovat logování a po ukončení simulace otevře program NAM a spustí vizualizaci simulace.

```
proc finish {} {
     global ns nf
     $ns flush-trace
     close $nf
     exec nam out.nam &
     exit 0
}
```

Potom zadáme, že simulace má skončit po 10 vteřinách simulačního času.

```
$ns at 10.0 "finish"
```

A na konec zapneme simulaci.

```
$ns run
```

Tímto máme vytvořen vzorový soubor, který můžeme potom využít k tvorbě dalších simulací.

2.1.5.2. Jednoduchý skript

Vytvoření dvou uzlů, propojení mezi nimi a posílání dat.

Jako první vytvoříme dva uzly pomocí příkazu *$ns node* a nastavíme jim jména na *uzel1* a *uzel2*.

```
set uzel1 [$ns node]
set uzel2 [$ns node]
```

Dále je potřeba vytvořit spoj mezi těmito uzly a to pomocí příkazu *$ns duplex-link*.

```
$ns duplex-link $uzel1 $uzel2 10Mb 8ms DropTail
```

Tímto příkazem simulátoru říkáme, že je třeba vytvořit duplexní spojení mezi uzly 1 a 2 s rychlostí 10Mb/s, zpožděním 8ms a frontou typu droptail.

Teď máme vytvořenu topologii, která ovšem nic nedělá. Aby bylo možné po topologii přenášet nějaká data, musíme si na uzlech vytvořit agenty. V Ns-2 bývají data vždy přenášena z jednoho agenta na druhého. Dalším krokem bude tedy tvorba agenta na uzlu1, který bude posílat data na uzel 2, který bude data přijímat.

```
#Create a UDP agent and attach it to node uzel1
set udp0 [new Agent/UDP]
$ns attach-agent $uzel1 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1000
$cbr0 set interval_ 0.01
$cbr0 attach-agent $udp0
```

Teď je vytvořen agent na uzlu1, který generuje konstantní CBR provoz. Bude vysílat každých 0,01 vteřiny paket o velikosti 1000 bajtů.
Dále vytvoříme agenta na uzlu 2, který bude generovaná data přijímat.

```
set null0 [new Agent/Null]
$ns attach-agent $uzel2 $null0
```

Ještě propojíme tyto dva agenty.

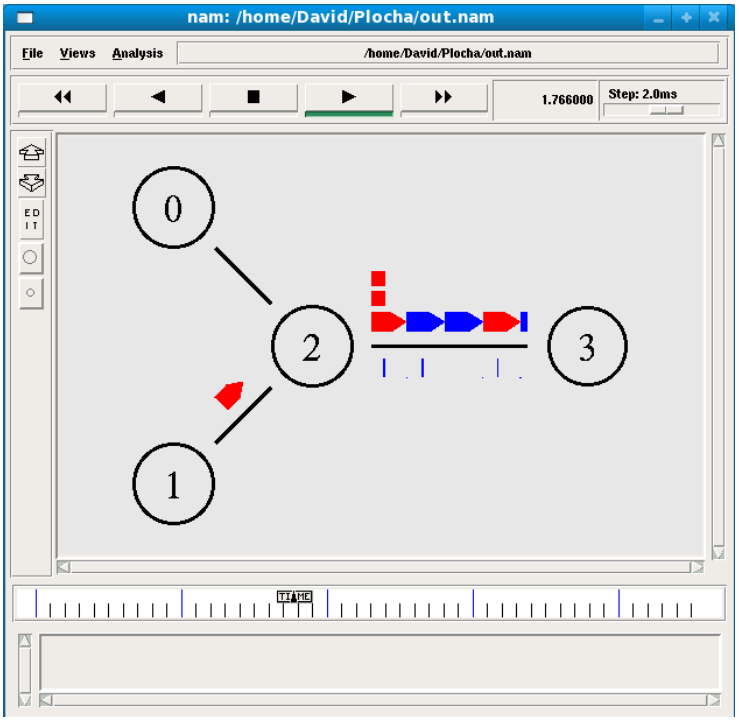
```
$ns connect $udp0 $null0
```

A zadáme čas vysílání.

```
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "$cbr0 stop"
```

Tímto máme vytvořen skript pro jednoduchou topologii dvou uzlů, kde jeden generuje data a druhý je přijímá.

Obr. 2: Nam vizualizace jednoduchého skriptu.



2.1.5.3. Agenti v NS2

Agenti v simulátoru NS2 zastupují koncové body na síti, které jsou schopny přijímat nebo odesílat data, generují provoz na síti. Agenti musí být připojeni na některý uzel v síti. Agenti jsou částečně implementováni v C++ a částečně v OTcl. Soubory pro agenty v C++ jsou umístěny v *~ns/agent.cc* a *~ns/agent.h* a pro agenty v OTcl v *~ns/tcl/lib/ns-agent.tcl*.

2.1.5.3.1. Stavy Agentů

Stavy se v agentech používají ještě předtím, než se paket odešle k příjemci pro nastavení jejich vlastností. Přehled jednotlivých stavů agentů je uveden v tabulce 1.

Více informací o nastavování stavů agentů je dostupné v manuálu na stránkách:

http://www.isi.edu/nsnam/ns/doc/node100.html

2.1.5.3.2. Protokoly Agentů

Agenti jsou na síti také schopni zastupovat jednotlivé komunikační protokoly. Ty nejpoužívanější jsou vypsány v tabulce 2.

Tab. 1: Stavy agentů

Stav	Funkce
addr	Adresa zdrojového agenta
dst	Adresa cílového agenta
size	Velikost paketu
type	Druh paketu
prio	Priorita paketu
flags	Označení paketu pomocí flagu

2.1.5.3.3. Základní příkazy pro agenty

`ns_ attach -` agent uzel agent - připojení agenta k uzlu
`$agent port –` určení zdrojového portu
`$agent dst-port –` určení cílového portu
`$agent attach-app s_type –` připojení aplikace k agentovi
`$ns_ connect zdroj cíl -` nastavení propojení mezi agenty
`$agent attach-trace file –` připojení souboru pro zachytávání komunikace

Tab. 2: Protokoly agentů

Protokol	Funkce
TCP/FullTCP	Kompletní obousměrná komunikace pomocí protokolu TCP
UDP	UDP protokol
RTP	Přijímač a vysílač RTP paketů
LossMonitor	Slouží pro generování a monitorování poruch a ztrát na síti
Message/Prune	Slouží pro přenos směrovacích informací
Null	Slouží pro likvidaci paketů
RtProto/DV	Směrovací protokol distance-vector

2.1.5.3.4. Vytvoření jednoduchého agenta

```
set tcp [new Agent/TCP]
      # nový tcp agent pro odesílání
set sink [new Agent/TCPSink]
      # nový tcp agent pro příjímání
$ns attach-agent $uzel0 $tcp
      # připojení tcp agenta pro odesílání dat na uzel 0
```

```
$ns attach-agent $uzel3 $sink
      # připojení  sink agenta pro příjem dat na uzel 3
ns connect $tcp $sink
      # vytvoření tcp spojení
set ftp [new Application/FTP]
      # vytvoření nové TCP aplikace
$ftp attach-agent $tcp
      # připojení aplikace na agenta pro odesílání
$ns at 1.2 "$ftp start"
      # start odesílání po 1.2 s od startu simulace
```

2.1.6. LAN sítě v Ns-2

LAN sítě jsou daleko složitější než doposud předváděné spoje typu konec – konec. Proto se pro LAN sítě v TCL skriptu využívá nový typ uzlu. Jedná se o uzel typu *LanNode*. Konfigurace těchto uzlů se může provádět v souborech *tcl/lan/vlan.tcl, tcl/lan/ns-ll.tcl, tcl/lan/ns-mac.tcl* ve složce *složka instalace*/ns.

Příklad vytvoření jednoduché sítě LAN mezi dvěma uzly na spojové vrstvě s frontou typu drop-tail a protokolem pro přístup k médiu CSMA/CD .

```
$ns make-lan "$n1 $n2" $bw $delay LL Queue/DropTail Mac/Csma/Cd
```

2.1.6.1. Vrstvy v LAN

Simulátor Ns-2 podporuje pro sítě LAN simulaci spodních tří vrstev. Jedná se o vrstvy fyzickou, spojovou a síťovou. Nejnižší fyzická vrstva obsahuje komponenty *Channel* a *Classifier/Mac*.

Komponenta *Chanel* zajišťuje simulaci přenosového média, komponenta *Classifier/Mac* se stará o komunikaci s vyšší vrstvou. Prostřední spojová vrstva se stará o propojování a přenos dat, řeší také kolize, které vznikají na přenosovém médiu. Nejvyšší síťová vrstva obsahuje dvě komponenty *Queue*, který pracuje jako zásobník a ukládá čekající pakety, které není možné okamžitě odeslat a *LL*, který se stará o komunikaci s nižší vrstvou a na druhou stranu o komunikaci s uzly. Nad těmito vrstvami jsou uzly, které pomocí těchto vrstev mezi sebou komunikují.

2.1.6.2. Chanel class

Třída *Chanel* slouží pro simulaci přenosového média, na němž se dá simulovat určité zpoždění. *Chanel class* také dokáže rozpoznat kolize na médiu a informuje o nich vyšší spojovou vrstvu a její komponentu *MAC*, která potom kolizi řeší.
Příklad vytvoření nového propojení se zpožděním 4μs.

```
set channel_ [new Channel]
      $channel_ set delay_ 4us
```

2.1.6.3. MAC class

Komponenta Spojové vrstvy zajišťuje adresování pomocí MAC adres a přístup na médium.

2.1.6.4. LL (link-layer) class

Komponenta síťové vrstvy, která se stará o simulaci propojovacích protokolů. Další důležitou službou je mapování MAC adres na IP adresy a směrování paketů.
Příklad nastavení LL (link-layer)

```
set ll_  [Jmeno LL] # Vytvoření vrstvy
set ifq_ [Dová fronta/DropTail] # použití fronty drop-tail
$ll_ lanrouter [new LanRouter $ns $lan] #nasavení adresy lan routeru
      $ll_ set delay_ $delay          # nastavení zpoždění
      $ll_ set bandwidth_ $bw         # nastavení datového toku
```

2.1.6.5. LanRouter class

V každé LAN síti může být defaultně 1 směrovač. Tento směrovač se vytváří při inicializaci simulace a jeho adresa je zanesena do všech uzlů v síti, tudíž není problém najít next-hop pro kterýkoliv paket v síti.
Příklad, jak směrovač hledá next-hop:

```
int LanRouter::next_hop(Packet *p) {
   int next_hopIP;
   if (enableHrouting_) {
         routelogic_\>lookup_hier(lanaddr_,adst,next_hopIP);
   } else {
         routelogic_\>lookup_flat(lanaddr_,adst,next_hopIP);
   }
```

2.1.6.6. Směrování

Pro směrování v simulátoru Ns2 se dají použít 4 různé možnosti:

- Unicast Routing
- Multicast Routing
- Network Dynamics
- Hierarchical Routing

Unicast routing – Zadaná data se přenášejí vždy pouze mezi dvěma zadanými uzly (hosty).

Multicast routing – Data se zasílají z jednoho zdrojového uzlu do více cílových uzlů.

Network Dynamics – Používá se, pokud se při běhu simulace přidávají a odebírají komunikační uzly. Simulační síť se během simulace mění.

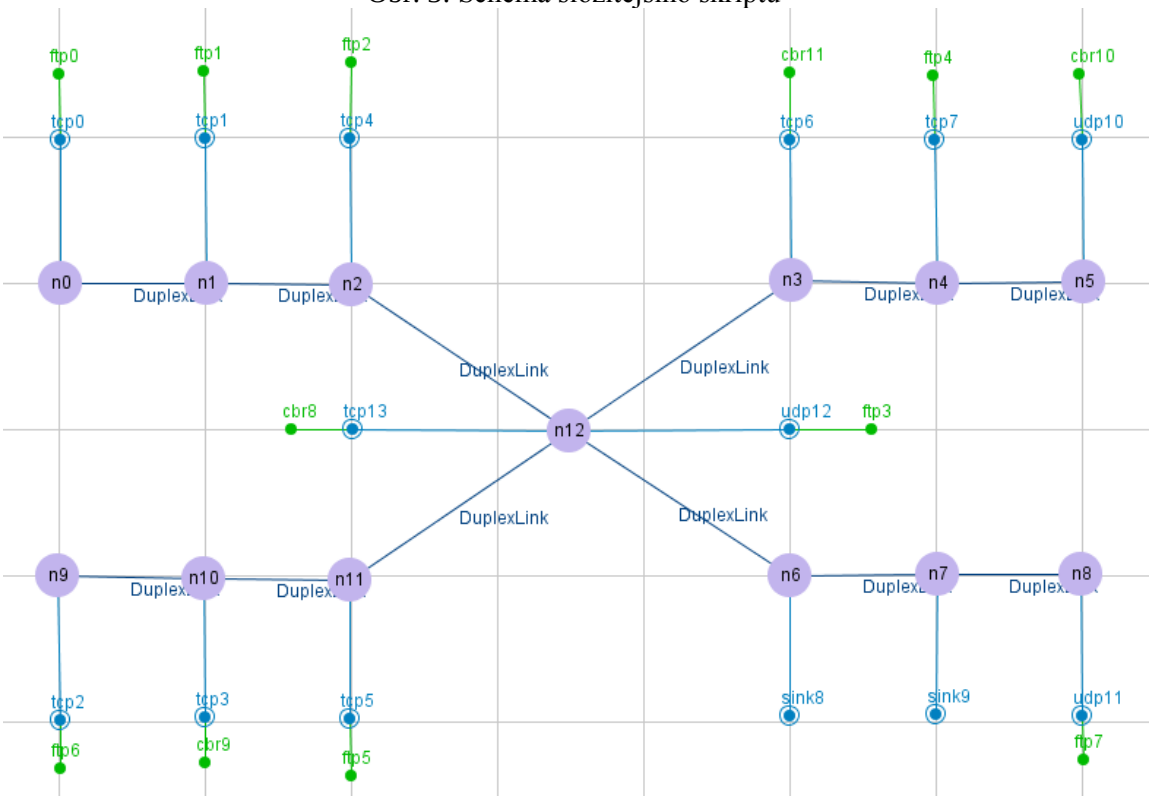
Hierarchical Routing - Celá simulační síť je rozdělena na více nezávislých oblastí, v každé se komunikuje samostatně, komunikace mezi jednotlivými oblastmi probíhá pomocí hraničních směrovačů.

Potřebné informace jak používat jednotlivé druhy směrování a jejich nastavení a vlastnosti jsou dostupné v manuálu na stránce: http://www.isi.edu/nsnam/ns/doc/node309.html

2.1.7. Ukázka složitějšího skriptu

V této simulaci je ukázáno použití jednotlivých spojů, různé přenosové rychlosti, a generování různých druhů provozu. Okomentovaný TCL skript je dostupný v příloze B.

Obr. 3: Schéma složitějšího skriptu



2.1.8. Bezdrátové sítě

Bezdrátové sítě v Ns-2 využívají podobné prostředky jako klasické drátové sítě. Těmito prostředky jsou bezdrátové uzly, na tyto uzly se potom připojují agenti a ti potom mohou generovat provoz. V bezdrátových sítích se nepoužívají klasické spoje mezi uzly, ale každému uzlu je možno nastavit dosah signálu. Pokud se v této oblasti nachází oblast signálu dalšího uzlu, mohou tyto uzly mezi sebou komunikovat. Veškeré bezdrátové uzly se také mohou pohybovat

v trojrozměrném prostoru. Tento pohyb se zadává pomocí výchozí pozice, a dále pozic, na kterých se má uzel nacházet v zadaných časových úsecích.

2.1.8.1. Směrování v bezdrátových sítích

Simulátor Ns-2 má implementovány 4 možnosti směrování pro bezdrátové sítě. Jsou to:

- Destination-Sequenced Distance Vector routing
- Dynamic Source Routing
- Ad hoc On-Demand Distance Vector Routing
- Temporally-Ordered Routing.

Destination-Sequenced Distance Vector routing - Uzel trvale udržuje kompletní směrovací informaci o celé síti, v okamžiku potřeby je dostupná cesta, ale hrozí zahlcení sítě přenášenými směrovacími informacemi.

Dynamic Source Routing, Ad hoc On-Demand Distance Vector Routing – Cesta k cíli se hledá, až když se skutečně potřebuje. Uchovávají se pouze aktivní směrovací informace, zpoždění při hledání cesty. Používá se pro větší sítě s častými změnami uzlů.

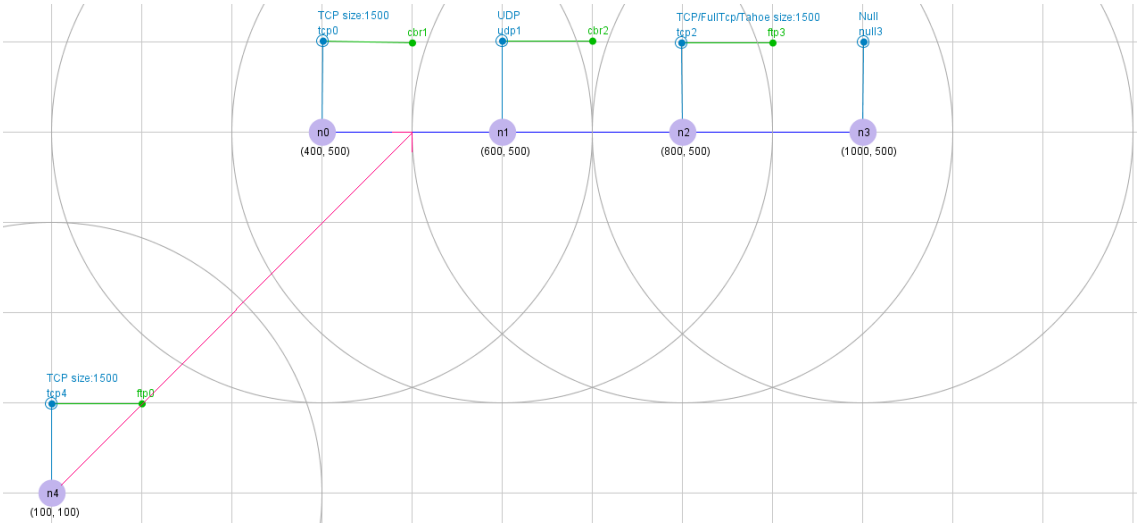
Temporally-Ordered Routing – Každý uzel dostane přiřazenou určitou výšku, směruje se pomocí těchto relativních výšek, nehledá se nejkratší, ale nejlevnější cesta tak, aby data tzv. „tekla z kopce“. Z vyšší výšky se data předávají na uzly s menší výškou.

Více o jednotlivých směrovacích metodách a jejich nastavení a použití je v manuálu na webových stránkách: http://www.isi.edu/nsnam/ns/doc/node179.html

2.1.8.2. Ukázka bezdrátového skriptu

Tento skript obsahuje 5 uzlů, 4 jsou umístěny v řadě a pátý se k nim přibližuje. Některé uzly generují provoz, jak TCP tak UDP. Jeden uzel je nečinný. Schéma rozložení sítě je vidět na obrázku. V příloze A je přiložen i okomentovaný TCL skript simulace.

Obr. 4: Schéma bezdrátové simulace



2.1.9. Satelitní sítě

Při simulaci satelitních sítí se dají používat dva typy satelitů. Jsou to buď satelity geostacionární, které se při pohledu ze země tváří jako by byly pořád na stejném místě, mají stejnou oběžnou rychlost jako je otáčení Země, nebo satelity na nízkém orbitu, které se vzhledem k pohledu ze Země pohybují.

Satelity se umisťují na oběžnou dráhu pomocí přesné pozice vůči rovníku a osám x, y a z. Použití satelitních sítí je podobné jako u bezdrátových sítí. Každý satelit pokrývá určitou část povrchu signálem. Komunikace může také probíhat navzájem mezi satelity na oběžné dráze. Při používání satelitních sítí se simuluje hlavně velké zpoždění, kterého se dosahuje kvůli dlouhým bezdrátovým přenosům mezi povrchem a satelitem. Satelitní sítě se v TCL skriptu samozřejmě dají propojit s bezdrátovými a drátovými sítěmi umístěnými na povrchu.

2.2. Simulátor Ns-3 (Network Simulator 3)

2.2.1. Úvod

Síťový simulátor Ns-3 využívá pro simulaci diskrétních událostí, je open source a jeho vývoj začal v roce 2006. Je to přímý nástupce simulátoru Ns-2. Simulátor Ns-3 je určen hlavně pro výzkum a vzdělávání, zatímco Ns-2 se dá nasadit prakticky kdekoliv. Simulátor Ns-3 je zatím pořád ve stádiu vývoje, proto se nedoporučuje jej nasazovat na větší projekty.

Program Ns-3 je tvořen v programovacím jazyku C++ a pro zápis skriptů obsahujících zadanou simulaci využívá jazyka Python. Simulátor nemá grafické rozhraní podobně jako Ns-2. Simulace se zadávají přímo z příkazového řádku, výsledky se ukládají do souboru. Tyto soubory se potom dají dále vizualizovat. Simulátor Ns-3 běží na systému Linux, pod systémem Windows se také dá spustit, ale to je třeba mít program pro simulaci Linuxového prostředí například Cygwin, viz. http://www.cygwin.com/ .

Hlavní webová stránka projektu Ns-3, dají se zde najít základní informace, stáhnout aktuální verze programu a získat nápovědu: http://www.nsnam.org

Stránka obsahující dokumentaci k programu, dají se zde najít i dokumenty týkající se architektury systému: http://www.nsnam.org/documents.html

Stránka na Wikipedii nabízející množství dalších odkazů ohledně projektu Ns-3: http://www.nsnam.org/wiki/index.php/Main_Page

Stránka obsahující zdrojový kód aplikace, lze se sem přihlásit pomocí kanálu RSS nebo Atom a odebírat průběžně změny ve zdrojovém kódu: http://code.nsnam.org/

2.2.2. Podporované protokoly

Simulátor Ns-3 dokáže simulovat činnost různých síťových protokolů. Jejich přehled je dostupný v tabulce 3.

Tab. 3: Podporované protokoly

Protokol	Funkce
TCP	Kompletní podpora TCP protokolu
UDP	Kompletní podpora UDP protokolu
IP	IP Protokol na 3. vrstvě
ARP	ARP protokol na 3. vrstvě
Jednotlivé aplikace	Využívají jednotlivé protokoly
Null	Slouží pro likvidaci paketů

2.2.3. Stažení a instalace Ns-3

Jako první je třeba si stáhnout instalační soubory z webu výrobce:

http://www.nsnam.org/download.html

Jakmile budeme mít staženy instalační soubory, je třeba je rozbalit a spustit instalaci. Nejprve se musíme přepnout do složky, kam jsme instalační soubory rozbalili, a potom spustíme instalaci pomocí příkazů:

```
./waf -d debug configure
```

Tento příkaz otestuje naše prostředí a zjistí, jestli je možně simulátor Ns-3 nainstalovat. Pokud tento test proběhl bez problémů, můžeme pokračovat v instalaci zadáním příkazu:

```
./waf
```

Tento příkaz již nainstaluje program do vašeho počítače. Po dokončení instalace by se mělo objevit hlášení, že vše proběhlo bez problémů.

```
Compilation finished successfully
```

Pokud proběhne instalace bez obtíží, je nutné ještě před spuštěním prvního skriptu se simulací sítě provést testy. Ty se spouštějí pomocí příkazu:

```
./waf check
```

Pokud i testy proběhly bez problémů, můžeme spustit první ukázkový skript. Ten je dodán přímo od výrobce. Spouští se příkazem:

```
./waf --run hello-simulator
```

Výstupem příkazu by mělo být:

```
Hello Simulator
```

Pokud tento skript proběhne bez problémů, můžeme si být jisti, že instalace proběhla dobře a simulátor Ns-3 je nainstalován a plně funkční.

2.2.4. Tvorba skriptu

Pro tvorbu skriptů se používá skriptovací jazyk python v kombinaci s programovacím jazykem C++. Skripty můžeme psát a upravovat v jakémkoli textovém editoru. Každý skript se skládá z několika modulů. Každý modul má přesně určenou svou funkci. V první části skriptu se pomocí modulů provádí určitá nastavení a v druhé části se nastavuje prostředí a chování simulace.

2.2.4.1. Modul Include

Každý skript by měl začínat modulem *include*. V něm se zadává, jaké externí knihovny budeme ve skriptu používat. Všechny knihovny je možno nalézt ve složce *složka instalace/ns3*. Funkce jednotlivých knihoven, jejich vazby a potřeby použití jsou uvedeny na internetových stránkách programu Ns-3.

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
```

2.2.4.2. Modul Ns-3 Namespace

Další část, kterou by měl obsahovat každý skript, je modul *Ns-3 namespace*. V něm zadáváme, ve kterém jmenném prostoru bude program pracovat. Pro všechny skripty Ns-3 se používá deklarace *namespace ns3*.

```
using namespace ns3;
```

2.2.4.3. Modul Logging

Tato část se používá na povolení či zakázání ukládání informací o běhu skriptu. Pokud tento řádek vynecháme, nebudou se informace ukládat. Přesné využití a nastavení ukládání informací si lze přečíst na stránkách nápovědy.

```
NS_LOG_COMPONENT_DEFINE ("LogFile");
```

2.2.4.4. Modul Main Function

I v tomto skriptu, tak jako v každém jiném programu C++, je třeba definovat funkci main. Ta, jako ve všech jiných programech, slouží jako výchozí bod programu. Obsah této funkce je závislý od toho, co bude skript vykonávat. Přesný popis, co do funkce main zadat, je opět v dokumentaci.

```
int
   main (int argc, char *argv[])
```

2.2.4.5. Modul Topology Helpers

Tento modul se skládá z dalších podmodulů a nastavuje se zde rozložení síťových prvků a nastavení spojů mezi nimi. Pomocí tohoto modulu tvoříme fyzickou síť, na které poběží naše simulace.

2.2.4.5.1. NodeContainer

Pomocí tohoto podmodulu vytváříme síťové uzly. Ty mohou reprezentovat počítače, na které se dají přidávat různé funkce. Pomocí tohoto příkladu vytvoříme 2 uzly a přidáme je do balíčku *nodes*.

Prvním řádkem aktivujeme funkci *NodeContainer,* která slouží k tvorbě uzlů. Na druhém řádku zadáme, že si přejeme vytvořit dva uzly.

```
NodeContainer nodes;
      nodes.Create (2);
```

2.2.4.5.2. PointToPointHelper

Pomocí podmodulu *PointToPointHelper* vytváříme logické spojení mez dvojicemi uzlů. Těmto spojením můžeme potom nastavovat různé vlastnosti. Ve skutečném světě by to znamenalo do počítačů zapojit síťové karty a ty mezi sebou propojit pomocí kabelů. V příkladu tvoříme spojení mezi dvěma uzly s rychlostí 5 Mb/s a zpožděním 5 ms. Na prvním řádku se aktivuje funkce *PointToPointHelper*. Pomocí ní potom na druhém a třetím řádku nastavíme rychlost linky a zpoždění.

```
      PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate",StringValue "5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue ("5ms"));
```

2.2.4.5.3. NetDeviceContainer

Pomocí dvou předchozích modulů jsme si vytvořili dva volné uzly a drátové spojení. Pomocí podmodulu *NetDeviceContainer* teď tyto uzly propojíme pomocí tohoto předem vytvořeného spojení. Ve skutečnosti by to znamenalo zasunout do síťových karet kabel. Pomocí prvního řádku opět aktivujeme funkci *NetDeviceContainer* a na druhém řádku provedeme instalaci spojení. Jakmile se tohle provede, máme připraven funkční spoj mezi dvěma uzly, který pracuje se zpožděním 5 ms a rychlostí 5 Mb/s.

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

2.2.4.5.4. InternetStackHelper

Nyní máme připravenou jednoduchou síť, ale nemáme zatím nainstalovány žádné protokoly, na kterých bychom mohli pracovat. K tomu nám slouží podmodul *InternetStackHelper*, pomocí kterého přidáváme na uzly jednotlivé protokoly nebo celé balíčky protokolů. Na prvním řádku se aktivuje funkce *InternetStackHelper* a na druhém se na oba uzly přidá balíček internetových protokolů. Jsou to protokoly IP, TCP, UDP atd.

```
InternetStackHelper stack;
      stack.Install (nodes);
```

2.2.4.5.5. Ipv4AddressHelper

Aby bylo možno mezi uzly komunikovat, je třeba jim přiřadit určité adresy. Pomocí tohoto modulu se jednotlivým uzlům přiřazují adresy Ipv4. Na prvním řádku se aktivuje funkce *Ipv4AddressHelper* která přiřazuje adresy a na druhém řádku proběhne samotné přiřazení. Adresy

budou přiřazovány od adresy 10.0.0.0 pomocí masky 255.255.255.0. To znamená, že našim uzlům budou přiřazeny adresy 10.0.0.1 a 10.0.0.2

```
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.255.255.0");
 Ipv4InterfaceContainer interfaces = address.Assign(devices);
```

Nyní máme vytvořenou jednoduchou síť, která je schopna komunikovat. Dále budeme potřebovat nějaké aplikace, které budou na síti generovat provoz.

2.2.4.6. Aplikace

V programu Ns-3 jsou dva druhy aplikací klienti a servery reprezentováni dvěma moduly *UdpEchoServer* a *UdpEchoClient*. Oba slouží pro generování provozu na síti.

2.2.4.7. UdpEchoServerHelper

Funkce *UdpEchoServerHelper* slouží k umisťování serverových aplikací na jednotlivé uzly v síti. Serverové aplikace odpovídají klientským aplikacím na jejich dotazy a generují tak provoz na síti. Na prvním řádku příkladu je aktivována funkce *UdpEchoServerHelper* číslo v závorce znamená port, na kterém bude server poslouchat dotazy od klientů. Na dalším řádku se serverová aplikace připojí na uzel. Na dalších řádcích je zadán čas v sekundách, kdy se má začít poslouchat a kdy skončit. Čas je udáván vůči startu aplikace.

```
UdpEchoServerHelper echoServer (9);
 ApplicationContainer serverApps = echoServer.Install
      (nodes.Get (1));
 serverApps.Start (Seconds (1.0));
       serverApps.Stop (Seconds (10.0));
```

2.2.4.8. UdpEchoClientHelper

Funkce *UdpEchoClientHelper* slouží pro umisťování klientských aplikací na uzly. Klientské aplikace komunikují se serverovými aplikacemi a generují tak provoz na síti. Na prvním řádku příkladu se volá funkce *UdpEchoClientHelper* a přiřazuje se jí vzdálená adresa a vzdálený port, se kterým bude komunikovat. V tomto příkladě adresa 1 a port 9. Adresa 1 znamená, že se bude komunikovat s první IP adresou, která byla přiřazena ze zásobníku adres v předchozí části. V našem případě adresa 10.0.0.1. Na dalších řádcích se nastavují jednotlivé parametry komunikace. Velikost paketů, interval v jakém se bude vysílat a maximální počet paketů, které může klient odeslat během simulace.

```
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
  echoClient.SetAttribute ("MaxPackets", UintegerValue (100));
```

Dále je stejně jako u serveru nutno nastavit začátek a konec vysílání.

```
ApplicationContainer clientApps = echoClient.Install (nodes.Get
(0));
    clientApps.Start (Seconds (2.0));
    clientApps.Stop (Seconds (10.0));
```

2.2.5. Simulátor

Jakmile máme sestavenu topologii sítě, můžeme spustit samotnou simulaci pomocí příkazu:

```
Simulator::Run ();
```

Ta provede námi naplánované úlohy, v našem případě to bude spuštění a zastavení serveru a klienta a zasílání dat. Jakmile budou tyto úlohy provedeny, simulátor nám řekne, že simulace byla dokončena. Ještě však zbývá uklidit. To se provede pomocí příkazu:

```
Simulator::Destroy ();
     return 0;
```

2.2.5.1. Sestavování skriptu

Sestavení skriptu je jednoduché. Stačí jej nakopírovat do složky scratch.

```
složka instalace/ns-3-dev > cp examples/first.cc scratch/
```

A dále už jen sestavit pomocí *./waf*

```
složka instalace /ns-3-dev > ./waf
```

Jakmile máme skript sestaven bez problémů, můžeme spustit simulaci.

```
složka instalace /ns-3-dev > ./waf --run scratch/first
```

2.2.6. Směrování

Směrování se v simulátoru Ns-3 provádí pomocí modulu *internet-stack*. Simulátor Ns-3 dokáže provádět směrování pomocí více směrovacích protokolů uvnitř jednoho simulačního skriptu. V současné verzi simulátoru jsou implementovány dva různé druhy směrovacích protokolů. Jsou to: *class Ipv4StaticRouting*, který zajišťuje unicast a multicast směrování a *Optimized Link State Routing*, který se používá pro mobilní sítě a pracuje na principu algoritmu stavu linek.

2.2.6.1. Statické směrování

Simulátor Ns-3 používá Statické směrování *Ipv4StaticRouting* jako výchozí možnost pro každý simulační skript. Pomocí něj se dají ke každému uzlu pevně nastavovat unicast nebo multicast cesty.

2.2.6.2. Unicast směrování

Nastavování unicast směrování se provádí pomocí metod:

```
void Ipv4::AddHostRouteTo ()    //Slouží pro zadání cesty k jednotlivému hostu na síti
void Ipv4::AddNetworkRouteTo () //Slouží pro zadání cesty k síti
void Ipv4::SetDefaultRoute ()    // Slouží pro nastavování výchozí cesty
uint32_t Ipv4::GetNRoutes ()     //Slouží pro získání několika cest
Ipv4Route Ipv4::GetRoute ()      //Slouží k získání jedné specifické cesty
```

Přesný popis k jednotlivým metodám je dostupný v dokumentaci na adrese: http://www.nsnam.org/doxygen/index.html

2.2.6.3. Multicast Směrování

Slouží k zasílání dat z jednoho zdroje na více cílových adres. Tyto příkazy se používají pro zadání statické multicast cesty k jednotlivému uzlu.

```
void
Ipv4StaticRouting::AddMulticastRoute (Ipv4Address origin,
                      Ipv4Address group,
                      uint32_t inputInterface,
                           std::vector<uint32_t>
outputInterfaces);
```

2.2.6.4. Globální centralizované směrování

Globální centralizované směrování využívá pro svou funkci spoje CSMA bod-bod. V současné verzi není ještě plně implementováno.

2.2.6.5. Směrování Optimized Link State Routing (OLSR)

První dynamický směrovací protokol implementovaný v simulátoru Ns-3. Využívá se hlavně u simulací mobilních sítí. Následující příkazy povolí OLSR směrování.

```
olsr::EnableAllNodes ();              // Zapne OLSR na všech uzlech
olsr::EnableNodes(InputIterator begin,InputIterator end);
// Zapne OLSR na vybraných uzlech
olsr::EnableNode (Ptr<Node> node); // Zapne OLSR pouze na jednom uzlu
```

Dále je ještě potřeba spustit agenta pomocí příkazu `Start()` a nastavit hlavní zařízení pomocí `SetMainInterface()`.

Více informací a podrobnější nastavení je dostupné v manuálu na stránce:
http://www.nsnam.org/docs/release/manual.html#SEC120

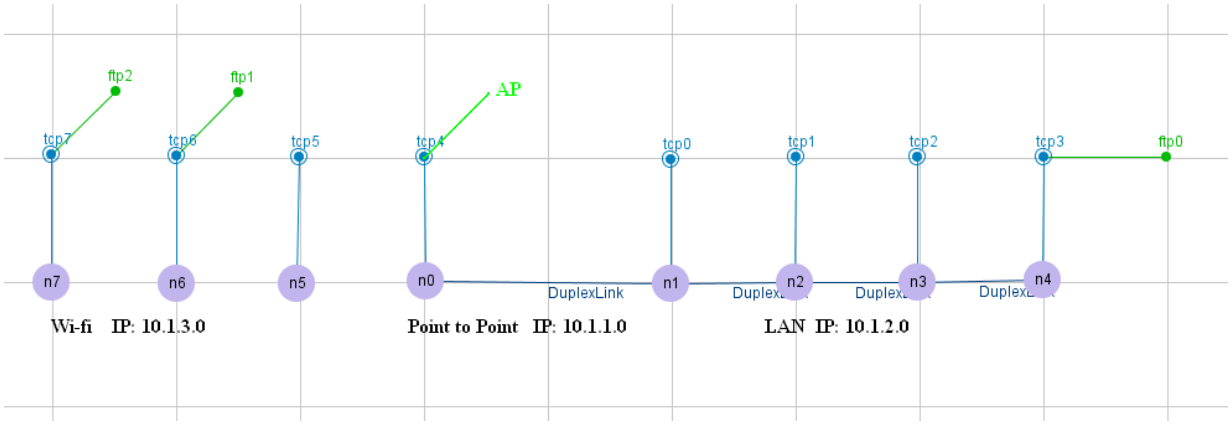
2.2.7. Wi-Fi sítě

Uzly v simulátoru Ns-3 mohou zastupovat různá zařízení. Například klasický ethernet adaptér, Wi-Fi zařízení nebo bluetooth modul. Wi-Fi zařízení se na Ns-3 uzel přidává pomocí příkazu *WifiNetDevice*. Každé zařízení *WifiNetDevice* je schopno simulovat Wi-Fi adaptér. Dají se mu nastavovat různé vlastnosti, například rychlost komunikace, zpoždění, dosah signálu a další. Hlavní věcí, která se dá u mobilních zařízení nastavovat, je jejich pohyb. Ten se nastavuje pomocí souřadnic X a Y, rychlosti a času, podobně jako u simulátoru Ns-2. Všechna tato nastavení i jednoduchý okomentovaný Wi-Fi skript jsou uvedeny v příloze C.

2.2.8. Ukázkový skript

V příloze C je dostupný okomentovaný ukázkový skript, který popisuje jednoduchou LAN síť tvořenou dvěma částmi. První část je bezdrátová, jsou na ní tři mobilní uzly a jeden přístupový bod, k tomu je potom připojena síť čtyř pevných stanic. Na této síti je simulována jednoduchá komunikace.

Obr. 5: Schéma ukázkového skriptu



2.3. Simulátor Ns-1 (Network Simulator 1)

2.3.1. Úvod

Síťový simulátor Ns-1 byl vyvinut společností Network Research Group na univerzitě v Berkley. Tento simulátor je předchůdcem simulátorů Ns-2 a Ns-3. V současné době je jeho další vývoj zastaven a je plně nahrazen programem Ns-2. Síťový simulátor Ns-1 se využívá stejně jako Ns-2 a Ns-3 pro simulaci diskrétních událostí v počítačových sítích. Pro tvorbu skriptů se používá jazyk TCL stejně jako v Ns-2. Ns-1 dokáže své výstupy zobrazovat pomocí programů XGraph a Nam stejně jako Ns-2.

Autoři doporučují používat novější verzi simulátoru Ns-2, protože má novou architekturu, která využívá MIT TCL objekty a jemnější rozložení tříd programu, tudíž simulace probíhá rychleji. Ns-2 dále obsahuje více funkcí a nastavení, která se při simulacích dají používat. Manuál pro verzi Ns-2 je také obsáhlejší a lépe zpracován.

2.3.2. Internet

http://www-nrg.ee.lbl.gov/ns/	// hlavní stránka
http://www-nrg.ee.lbl.gov/ns/man.html	//manuálové stránky
ftp://ftp.tns.lcs.mit.edu/pub/otcl/README.html	// MIT TCL objekty
ftp://ftp.ee.lbl.gov/ns	//stažení Ns-1

2.3.3. Instalace

Instalace simulátoru Ns-1 se provádí spuštěním skriptu.

```
./configure
```

Ten se nachází přímo ve složce, kterou jsme získali rozbalením instalačního archivu. Jakmile tento skript proběhne, je třeba spustit *make*, což by mělo vytvořit spustitelný soubor *Ns*. Pro bezchybnou instalaci je třeba používat TCL verze 7.5 a vyšší a překladač GCC 2.7.2 a novější. Jakmile proběhne instalace, je třeba ještě provést testy. Ty se spouštějí pomocí příkazu.

```
./ns test-suite.tcl tahoe1
```

Jakmile tento test proběhne korektně, je možné simulátor začít používat.

2.3.4. Závěr

Vzhledem k tomu, že simulátor Ns-1 je v dnešní době překonán a plně nahrazen simulátorem Ns-2 nebudu se v tomto textu věnovat přesnému vysvětlování a popisu jednotlivých funkcí. Veškerá funkcionalita, kterou obsahuje Ns-1, je totiž obsažena v simulátoru Ns-2. Skripty pro Ns-1 se píšou stejným způsobem jako pro Ns-2. Pokud bychom měli nějaké skripty jazyka TCL napsané pro simulátor Ns-1, je možné je bez problémů spustit v simulátoru Ns-2 se stejným výsledkem. Pokud porovnáme rychlost sestavování simulace a běhu, tak dojdeme k závěru, že u stejného skriptu, který byl původně určen pro simulátor Ns-1, dosahuje simulátor Ns-2 naprosto srovnatelných výsledků. Není tedy důvod pro používání zastaralého simulátoru Ns-1. Dokonce i hardwarové nároky obou simulátorů jsou velice podobné, jediný rozdíl je v použitém místě na disku. Zatímco simulátor Ns-1 zabírá přibližně 10 Mb, tak Ns-2 už zabírá přibližně 100 Mb, ale tento rozdíl je vyvážen vyšší funkcionalitou.

2.4. Srovnání simulátorů Ns-1, Ns-2 a Ns-3

Jako první z rodiny simulátorů Ns byl vydán v roce 1995 simulátor Ns-1. Na něj poté navázal v roce 2000 Ns-2 a v roce 2006 zatím poslední verze Ns-3. Verze Ns-1 a Ns-2 jsou velice podobné, pro svůj běh obě využívají TCL skripty. Simulátor Ns-3 už pracuje se skripty v jazyku python. Funkce všech tří simulátorů jsou velice podobné, všechny pracují na principu diskrétních událostí a ze zdrojového skriptu, ve kterém je popsán scénář simulace, sestavují simulační prostředí a v něm provádějí zadané úkoly. Se všemi simulátory se pracuje na úrovni příkazové řádky a výstupy všech simulátorů je poté možno zobrazovat pomocí přiložených programů. V dnešní době se doporučuje používat verzi Ns-2. Verze Ns-1 se už delší dobu nevyvíjí a byla plně nahrazena verzí Ns-2. A Ns-3 je zatím pořád ve stádiu vývoje, tudíž se může stát, že některý skript ještě nebude fungovat, protože ne všechny součásti simulátoru byly implementovány. V simulátoru Ns-3 je přidána podpora pro Bluetooth zařízení. Ve verzi Ns-2 se dá podpora Bluetooth doinstalovat, více na adrese: http://www.nsnam.org/doxygen/index.html A ve verzi Ns-1 chybí podpora Bluetooth zařízení úplně. Ve verzi Ns-1 je také velice malá podpora bezdrátových sítí Wi-Fi.

Nejvíce funkcí a možností simulace obsahuje simulátor Ns-2, který je nyní také doporučován jako hlavní zástupce všech těchto simulátorů jak pro vědecké, tak i profesionální použití. Verze Ns-1 byla totiž verzí Ns-2 přímo nahrazena a do verze Ns-3 se pořád přidávají nové funkce a možnosti simulace.

3. Síťový emulátor

Síťový emulátor je software, který slouží pro simulování chování síťového prostředí. Na malé síti je schopen generovat zpoždění, chybovost linek, ztrátovost a poškozování paketů takovým způsobem, aby se daly otestovat vyšší aplikace a jejich schopnost provozu na velkých sítích, kde jsou tyto problémy běžné. Mezi síťovým simulátorem a emulátorem je základní rozdíl. Zatímco simulátor vytváří virtuální síť, na které generuje provoz a různé stavy síťových prvků, tak emulátor pracuje s fyzickou sítí, kde pouze upravuje její chování. Síťové emulátory mají hlavní použití na testování síťových protokolů, aplikací a jejich chování v rozlehlých sítích bez nutnosti tuto velkou síť provozovat nebo se do ní připojovat.

V této práci budou rozebrány tři síťové emulátory. Jedná se o programy Net: Netem, Wanem a Wanulator. Všechny tyto emulátory jsou zdarma dostupné ke stažení na internetu.

3.1. Síťový emulátor Net:Netem

3.1.1. Úvod

Program Net:Netem poskytuje možnosti testování síťových protokolů, s jeho pomocí se dají simulovat rozsáhlé sítě. Současná verze nabízí nastavování délky zpoždění na lince, ztrátu, duplikaci a změnu pořadí paketů. Aktuální verze programu Net:Netem je obsažena v balíčku *Iproute2* a ten je přímo povolen ve většině současných Linuxových distribucí. (OpenSuse, Debian, Gentoo, Mandriva, Fedora, Ubuntu) Net:Netem tudíž není třeba stahovat z internetu a instalovat, je dostupný ihned po instalaci systému Linux.

Program Net:Netem se ovládá z příkazové řádky pomocí příkazu *tc*. Ten je součástí balíčku *Iproute2*. Příkaz *tc* využívá sdílené knihovny a data ze složky */usr/lib/tc*

Net:Netem se dá povolit přímo v jádře systému pod:

```
Networking -->

   Networking Options -->

     QoS and/or fair queuing -->

        Network emulator
```

3.1.2. Příklady použití

Na následujících řádcích bude ukázáno, jak se s programem Net:Netem pracuje a k jakým druhům simulací se dá použít. Veškeré příklady se provádí pomocí příkazového řádku a příkazu *tc*. Program Net:Netem nevytváří žádné výstupní soubory. Výsledky jeho práce se dají pozorovat pomocí síťového analyzátoru např. Wireshark.

3.1.2.1. Zpoždění v rozsáhlých sítích WAN

Na začátek ten nejjednodušší příklad, pomocí kterého pouze přidáme pevné zpoždění ke všem paketům odchozích z naší síťové karty. Ke všem paketům odcházejícím ze zařízení eth1 bude přidáno pevné zpoždění 125ms.

```
# tc qdisc add dev eth1 root netem delay 125ms
```

Nyní pomocí příkazu ping na jakýkoliv počítač můžeme zjistit nárůst zpoždění o 125ms. Nastavování zpoždění je závislé od taktu procesoru, ale většinou jde nastavovat po krocích 5 až 20 ms.

Ve skutečných sítích však není zpoždění konstantní, ale proměnné. Toho můžeme dosáhnout pomocí programu Net:Netem s příkazem *change*. Tento příklad bude generovat zpoždění 120ms +- 20ms

```
# tc qdisc change dev eth1 root netem delay 120ms 20ms
```

Anebo lze také použít příkaz *change*, pomocí kterého se dá nastavit velikost zpoždění, která bude závislá na předchozím zpoždění v procentech. Na tomto příkladu je to 120ms +- 20ms, ale další zpoždění se nebude lišit o více než 25%.

```
# tc qdisc change dev eth1 root netem delay 120ms 20ms 25%
```

3.1.2.2. Ztráta paketů

Pomocí programu Net:Netem se dá napodobovat ztrátovost paketů na linkách. Ta se udává v procentech a dá se nastavit od minimální hodnoty 2^{32} = 0.0000000232% až po 100%.

Na základním příkladu si předvedeme státovost 0,2%, to znamená, že z 1000 paketů budou pomocí náhodného výběru 2 zahozeny.

```
# tc qdisc change dev eth1 root netem loss 0.2%
```

Dále se taky dá přidat ztrátovost po větších blocích. Pomocí tohoto příkladu bude zahozeno 0,5% paketů a s pravděpodobností 25% se bude jednat o pakety jdoucí za sebou.

```
# tc qdisc change dev eth1 root netem loss 0.5% 25%
```

3.1.2.3. Duplikace paketů

Napodobování duplikace paketů pracuje na stejném principu jako ztrátovost paketů. To znamená, že zadáváme procentuální hodnotu, kolik paketů se má duplikovat. V následujícím příkladu bude duplikováno 10% paketů.

```
# tc qdisc change dev eth1 root netem duplicate 10%
```

3.1.2.4. Poškození paketů

Pomocí Net:Netem se taky dá napodobovat poškozování paketů. V praxi to znamená, že program vloží do náhodného offsetu v paketu bitovou chybu. Nastavování se opět provádí pomocí zadání, v kolika procentech paketů chceme chybu způsobovat.

```
# tc qdisc change dev eth1 root netem corrupt 0.5%
```

3.1.2.5. Změna pořadí paketů

V programu Net:Netem jsou dva příkazy pro změnu pořadí paketů. Jsou to buď *gap* nebo *reorder*.

Příkaz *gap* používá pevné pořadí a přeskupuje každý n-tý paket. V příkladu je ukázáno, že každý pátý paket bude odeslán okamžitě a všechny ostatní budou mít zpoždění 10ms.

```
# tc qdisc change dev eth1 root netem gap 5 delay 10ms
```

Další příkaz je *reorder*, ten zajišťuje zpoždění paketů zadávané v procentech. Na příkladu je vidět, že 25% paketů bude odesláno okamžitě a zbytek se zpožděním 10ms, což zajistí změnu pořadí a náhodné uspořádání paketů.

```
# tc qdisc change dev eth1 root netem delay 10ms reorder 25%
```

3.1.3. Závěr

Program Net:Netem nemá sice tolik možností jako pravý síťový simulátor, ale to, že je integrován ve většině linuxových distribucí a nemusí se instalovat, ho předurčuje k tomu, aby se s jeho pomocí prováděly jednoduší testy na jakémkoli počítači se systémem Linux a připojením k síti. Net:Netem poskytuje základní funkce a jednoduché ovládání přes příkazovou řádku. Bez potřeby učit se složitou syntaxi a psát simulační skripty nám dovoluje vyzkoušet, jak by vypadal provoz na velké síti se ztrátou, změnou pořadí a poškozováním paketů. Net:Netem není určen pro testování celých síťových řešení, ale spíš k tomu, abychom zjistili, jak se bude chovat ten daný protokol, aplikace nebo počítač ve velké síti, kterou jsem schopni napodobit pouze s naším počítačem a připojením do jakékoliv sítě.

3.2. Síťový emulátor WANulator

3.2.1. Úvod

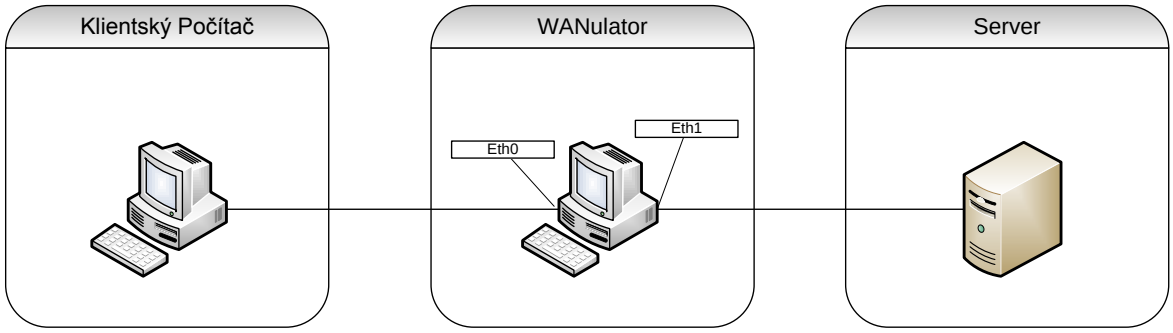
Pro testování byla zvolena aktuální verze 1.01.

WANulator je software, který slouží k napodobování chování různých softwarových prostředků v různých síťových prostředích. Pracuje jako prostředník mezi serverem a klientem, kde se připojí na linku pomocí dvou síťových karet. Jedna pro vstupní a druhá pro výstupní provoz. Program dokáže napodobit různé stavy sítí. Může způsobovat zpoždění na linkách, ztrátu paketů měnit rychlost linek a další. Program je volně dostupný ke stažení. Jeho hlaví výhodou je, že nepotřebuje žádný operační systém. Je totiž dodáván jako bootovací CD linuxu. Jediné, co je potřeba, je PC s přístupem do sítě.

3.2.2. Hlavní funkce

Obsahem bootovacího CD je systém Wanulator, dále je přiložen Wireshark pro záznam a analýzu síťového provozu, Netperf pro generování přídavného provozu na síti, ssh a Webový prohlížeč Firefox.

Obr. 6: Schéma připojení systému WANulator do sítě



Systém Wanulator dokáže:

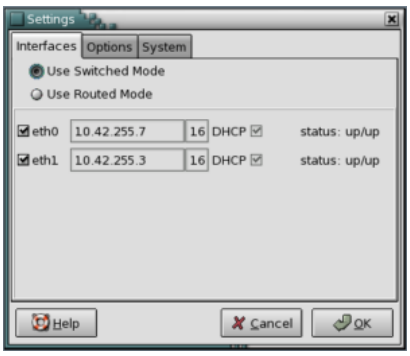
- Vkládat zpoždění
- Přidávat a odebírat paket z provozu
- Měnit pořadí paketů
- Poškozovat pakety
- Měnit přenosovou rychlost na linkách

Každá z těchto funkcí může být aplikována pouze na určitý druh paketů, např. pouze http provoz.

3.2.3. Příklady použití

Před prvním použitím programu je třeba jej nabootovat z CD. Po startu systému z CD bude zobrazen dialog pro konfiguraci síťových karet. Každé síťové kartě se dá nastavit IP adresa, nebo nechat adresy přidělit DHCP serverem. Dále se nastavuje, jestli budeme pracovat v přepínaném nebo směrovaném prostředí. Při použití Wanulatoru jako prostředníka mezi serverem a klientem použijeme přepínané prostředí.

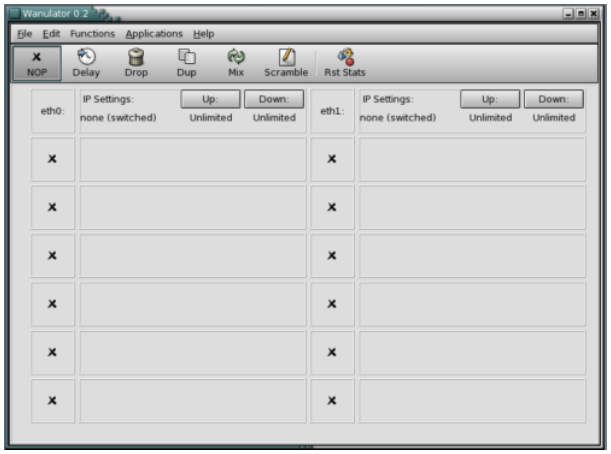
Obr. 7: Dialog pro konfiguraci síťových karet [1]



Jakmile proběhne tohle základní nastavení, přepneme se do hlavního okna programu.

Hlavní okno se skládá ze třech částí. Nahoře lišta s menu, pod ní výběr jednotlivých funkcí a na zbytku plochy jsou zobrazeny jednotlivé síťové karty a jejich nastavení. V horní části nastavení síťových karet se dá nastavovat jejich rychlost. Pro směr up jsou to možnosti 56, 128, 768, 1024 a 10 000 Kb/s, pro směr down to v aktuální verzi 1.01 ještě není implementováno. Pokud chceme aplikovat nějakou funkci, například vkládání zpoždění, dělá se to následovně: V liště výběru funkcí v horní části okna vybereme, kterou funkci budeme chtít aplikovat, potom klikneme na volný, nebo již použitý slot na síťové kartě. Při kliknutí na již použitý slot bude aktuální funkce deaktivována a nahrazena funkcí novou. Tímto bude tato funkce přiřazena, ale ještě nebude pracovat. Je třeba provést její nastavení.

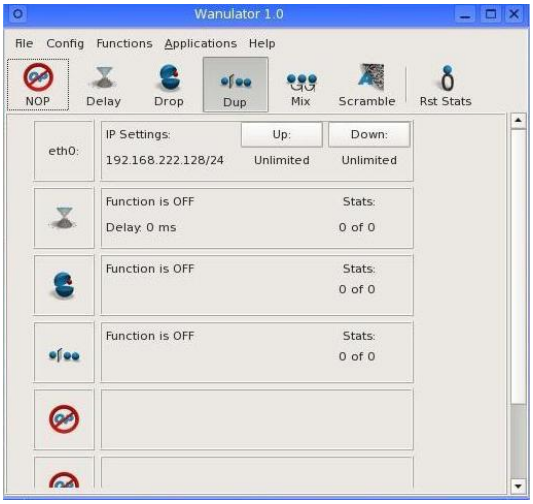
Obr. 8: Hlavní okno programu verze 0.2 [1]



[1] Obrázky převzaty z http://www.wanulator.de/

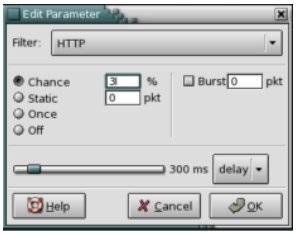
Na obrázcích č. 8 a 9 je vidět rozdíl při připojení jedné a dvou síťových karet. Ve verzi 0.2, obr. 8 je dialogové okno pro dvě síťové karty, ve verzi 1.0, obr. 9 je připojena pouze jedna.

Obr. 9: Hlavní okno programu verze 1.0 [2]



Nastavení funkcí se provádí v okně pro nastavování funkcí. Každá jednotlivá funkce má své vlastní dialogové okno a v něm se provádějí příslušná nastavení.

Obr. 10: Okno Nastavení funkcí [2]



Ostatní programy jako Firefox a Wireshark se spouštějí pomocí pravého kliknutí myší na ploše a vybráním dané aplikace ze seznamu.

3.2.4. Závěr

Program WANulator vyniká hlavně svou jednoduchostí a rychlostí práce. Není třeba nic instalovat, stačí nám pouze počítač, který má alespoň jednu síťovou kartu a je schopen bootovat z CD. Pro lepší simulaci je doporučeno používat dvě síťové karty, jednu pro provoz Up a jednu pro provoz Down. WANulator je určen hlavně pro testování chování síťových aplikací charakteru klient server. Poskytuje možnosti nastavení simulace chování těchto aplikací v rozsáhlých sítích s velkým zpožděním, státovostí paketů a s výpadky na linkách. Jeho možnost nastavování rychlosti linek také dokáže napodobit pomalou uživatelskou přípojku. WANulator není určen pro testy celých síťových řešení, ale pouze pro testy chování aplikací, které budou na síti pracovat.

[2] Obrázky převzaty z http://www.wanulator.de/

3.3. Síťový emulátor WANem

3.3.1. Úvod

Pro testování byla zvolena aktuální verze 1.2.1

WANem (Wan Area Network EMulation), jedná se o emulátor rozlehlých počítačových sítí, slouží pro testování chování síťových aplikací např. VoIP telefonie v různých síťových prostředích. Program je dodáván na bootovacím CD postaveném na linuxové distribuci Knoppix. WANem se spouští pomocí bootování z CD nebo pomocí programu VMware. Nedá se zatím nainstalovat na pevný disk, ale v dalším vydání se s tím počítá. WANem se spouští na jednom počítači připojeném do sítě. Nastavení se provádí z jiného počítače pomocí webového rozhraní. Dají se nastavovat zpoždění, rychlost linky, ztrátovost paketů a změna pořadí paketů.

3.3.2. Instalace a nastavení

Nejdříve je třeba spustit systém z bootovacího CD. Jakmile bude systém spuštěn, nastavuje se IP adresa. Ta se dá buď nechat přidělit DHCP serverem nebo se přidělí ručně. V dalším kroku se ještě zadá heslo pro vzdálený přístup pomocí SSH. Po dokončení tohoto nastavení je WANem přístupný pomocí webového rozhraní na adrese: *http://Zadana_IP /WANem*

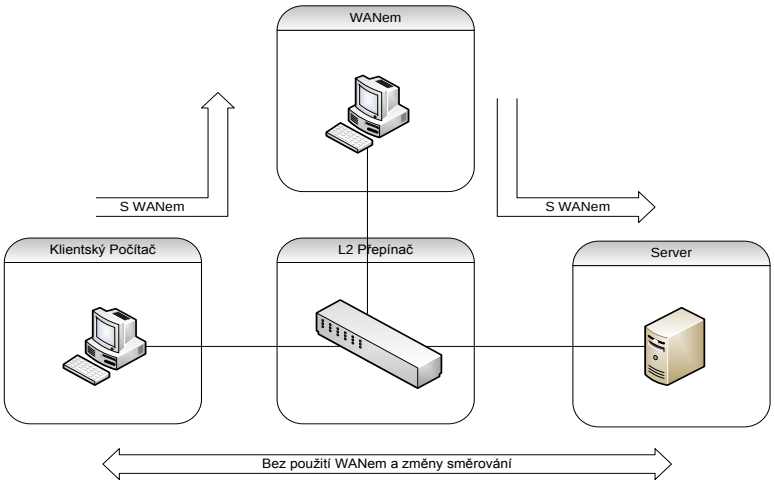
Další možnost přístupu je pomocí SSH, přihlašovacího jména „*perc*“ a hesla, které jsme si zadali.

V příloze D je možné si prohlédnout jednotlivé obrázky z průběhu instalace.

Další nastavení se provádí pomocí webového rozhraní z jiného počítače. Další nastavení závisí na rozložení síťových prvků, s kterými WANem pracuje. Existují dvě možnosti rozložení.

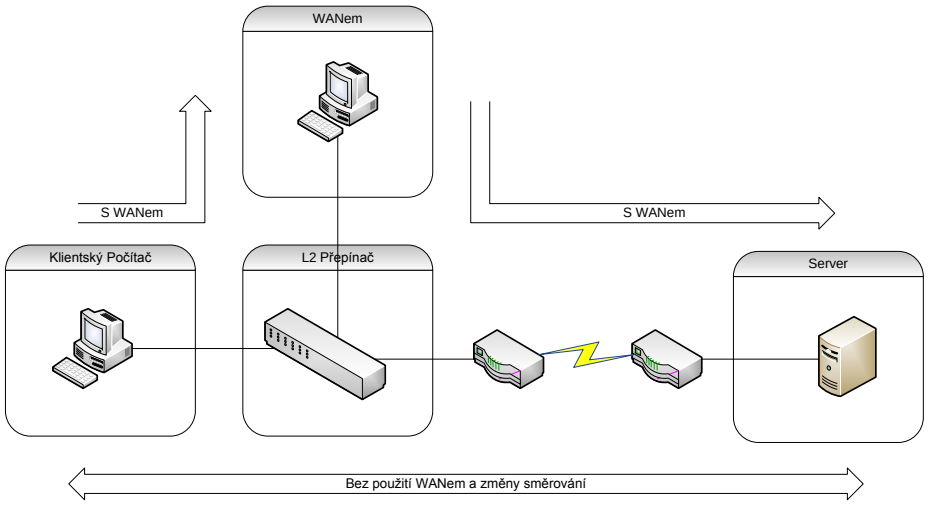
Buď jsou všechny komponenty umístěny v jedné podsíti a na cestě mezi klientem a serverem není žádný směrovač. V tomto případě nám stačí pouze na klienta a na server nastavit jako výchozí bránu adresu počítače kde běží WANem. Tato situace je znázorněna na obr. 11.

Obr. 11: Situace 1



Další možnost je, že je server umístěn v jiné síti za směrovačem. V tomto případě je třeba v programu WANem nastavit adresy přeskoků, které bude využívat. V tomto případě se opět klientovi nastaví jako výchozí brána adresa počítače, kde běží WANem a na serveru se nenastavuje nic. WANem zde slouží jako NAT a odpovědi ze serveru jsou směrovány na počítač s WANem a ten je potom předává ke klientovi. V tomto případě však spojení musí vždy inicializovat klient. V opačném případě by komunikace probíhala přímo bez použití emulátoru.

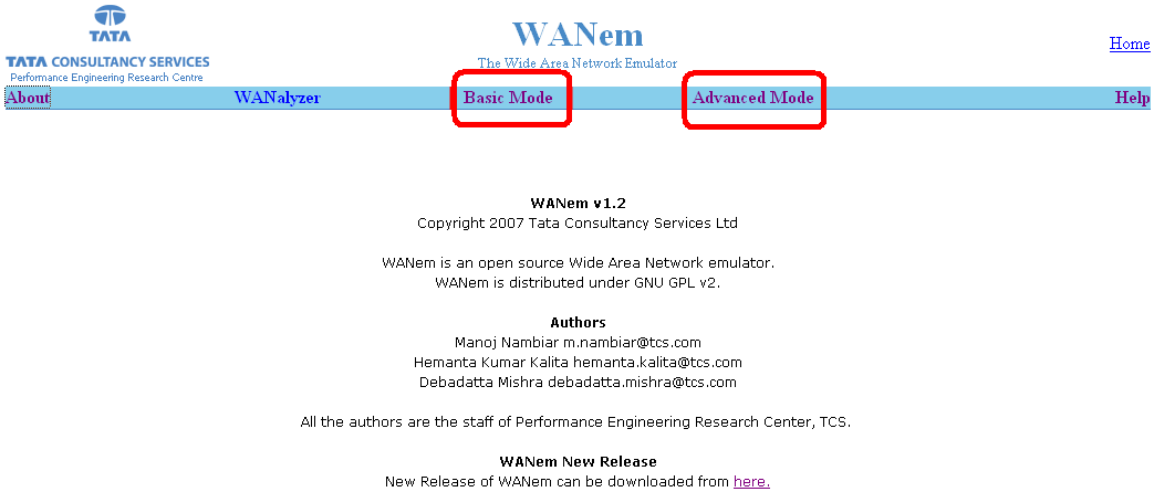
Obr. 12: Situace 2



3.3.3. Použití

Emulátor Wanem se dá používat ve dvou rozdílných módech. Jedná se o módy Basic a Advanced. Přepínání se provádí přímo na hlavní obrazovce.

Obr. 13: Wanem Hlavní obrazovka



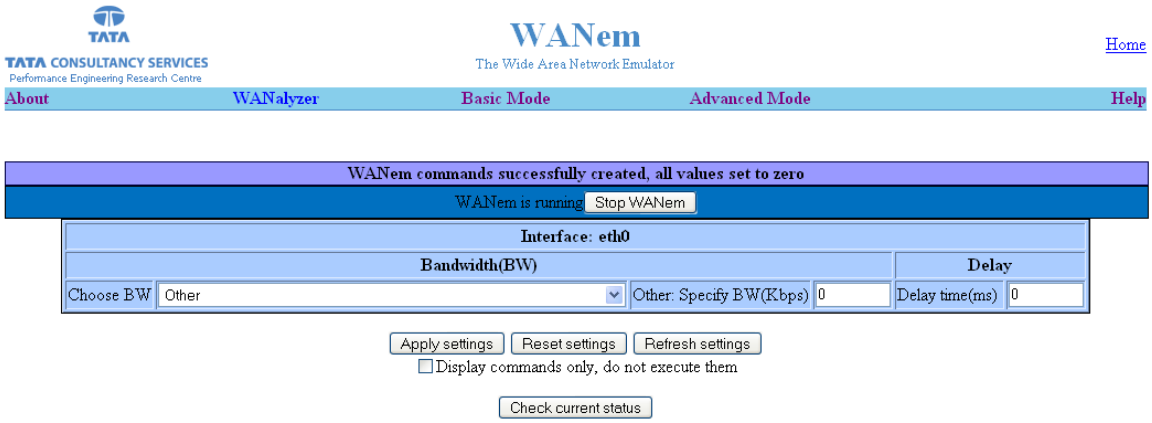
3.3.3.1. Basic Mode

V Basic Mode se dá nastavovat zpoždění a rychlost linky. Zpoždění se dá nastavovat v milisekundách. Změna se aplikuje na obě linky, tudíž pokud třeba zadáme zpoždění 50ms, výsledkem bude zpoždění 100 ms, protože poprvé bude zpoždění aplikováno na lince k serveru a potom znova na lince ke klientovi. Proto je potřeba zadávat vždy polovinu požadovaného zpoždění. Rychlost linky se dá nastavovat buď ručně, nebo pomocí předdefinovaných hodnot. Změny se dají otestovat pomocí příkazu ping z klienta na server.

Ukázka příkazu Ping a změny zpoždění, 50ms na každé lince

```
Pinging 192.168.1.2 with 1500 bytes of data:
Reply from 192.168.1.3: bytes=1500 time=9ms TTL=12
Reply from 192.168.1.3: bytes=1500 time=7ms TTL=12
      -> Aplikována změna
Reply from 192.168.1.3: bytes=1500 time=108ms TTL=12
Reply from 192.168.1.3: bytes=1500 time=110ms TTL=12
```

Obr. 14: Obrazovka Wanem Basic Mode



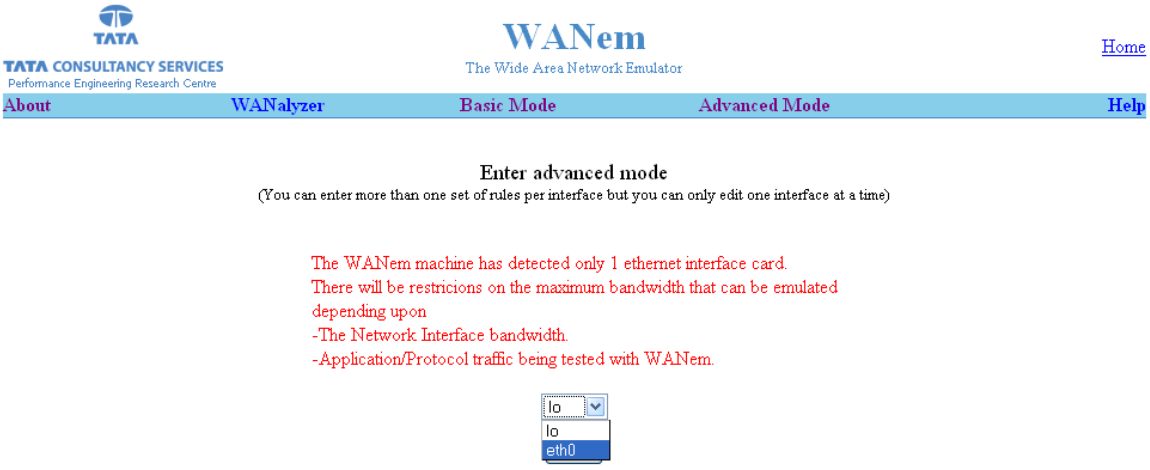
3.3.3.2. Advanced Mode

Když aktivujeme mód Advanced na hlaví obrazovce pomocí tlačítka advanced, je třeba vybrat rozhraní, se kterým budeme pracovat. Pokud bychom měli více rozhraní, můžeme mít provedeny změny na více rozhraních, ale pouze na jednom je můžeme v dané chvíli upravovat.

Jakmile máme vybráno rozhraní, můžeme začít provádět změny. V Advanced mode se dají nastavovat veškeré funkce programu. Měnit rychlost linky, nastavovat zda se jedná o symetrickou síť tj. zda zpoždění přidávat pouze jednou nebo dvakrát tak, jako v basic mode. Dále se dají nastavit velikost zpoždění, ztrátovost, duplikace, změna pořadí a poškození paketů. U každého nastavení jsou ještě další možnosti. Veškerá nastavení lze provádět jak jednotlivě, tak i dohromady. Ještě se také dá nastavit možnost úplného rozpojení a opětovného spojení linky. Ve spodní části okna se také dá nastavit to, že tato pravidla se budou aplikovat pouze na některé pakety podle IP adres a portů.

V příloze E je možno si prohlédnou jednotlivé výstupy příkazu ping pro jednotlivá nastavení.

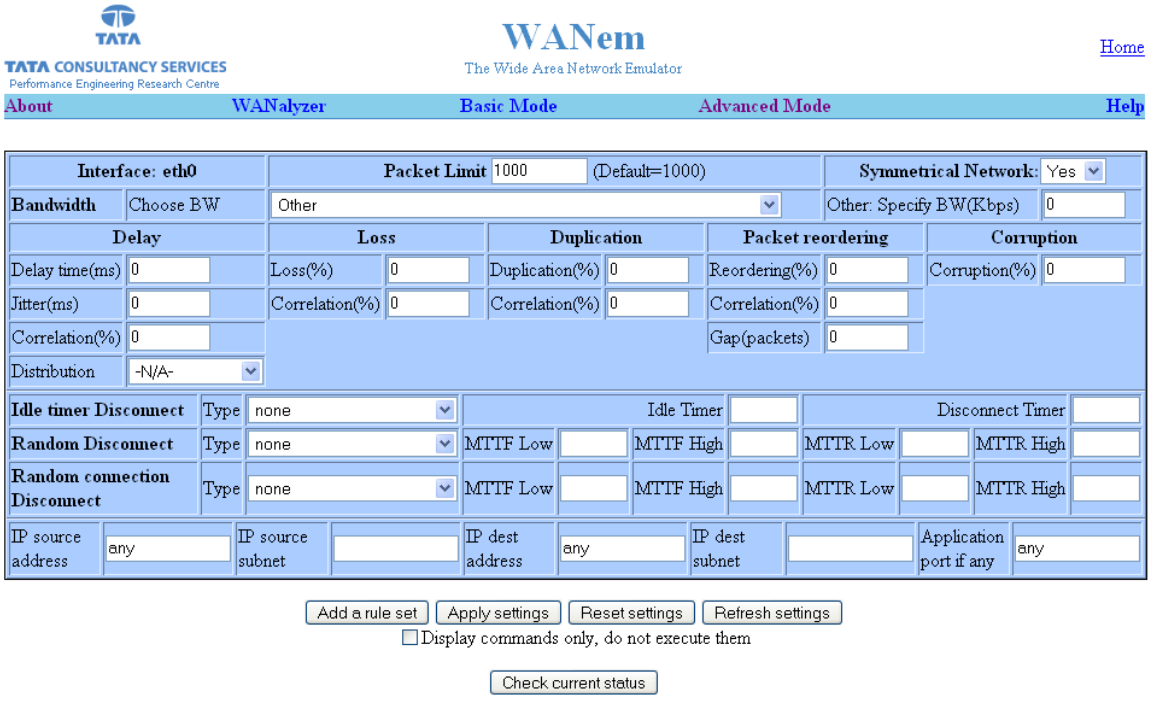
Obr. 15: Obrazovka Wanem Enter Advanced Mode



3.3.4. Závěr

WANem je síťový emulátor, sloužící pro zjišťování chování různých síťových aplikací v rozsáhlých sítích. Mezi jeho hlavní výhody patří možnost vzdáleně jej nastavovat pomocí webového rozhraní. Dokáže také nezávisle pracovat na více síťových kartách a změny aplikovat pouze na vybranou komunikaci. Tyto možnosti a jednoduchost použití jej předurčují jako nejlepší volbu ze všech testovaných emulátorů.

Obr. 16: Obrazovka Wanem Advanced Mode



3.4. Srovnání síťových emulátorů

Všechny emulátory zmíněné v této práci měly velice podobné funkce. Dva z nich, WANem a WANulator, jsou dodávány na live CD, ke svému běhu nepotřebují operační systém. Poslední Net:Netem je součástí operačních systémů Linux. Takže s instalací těchto programů není vůbec žádný problém. WANem a WANulator pracují na samostatných počítačích připojených do sítě, proto je před prvním použitím třeba tyto stroje jednoduše nastavit a změnit směrování v síti aby komunikace probíhala přes tyto stroje. Zatímco Net:Netem se provozuje přímo na počítači, na kterém se pracuje, tudíž není třeba provádět žádné změny. WANem a WANulator se ovládají přes grafické uživatelské rozhraní, zatímco Net:Netem se ovládá pouze pomocí příkazové řádky. Fungování a principy jednotlivých programů jsou vysvětleny přímo v částech textu věnovaných těmto programům. Všechny emulátory obsahují základní funkcionalitu, mezi kterou patří nastavování zpoždění, ztráta, duplikace, poškozování a změna pořadí paketů. K těmto základním funkcím přidává každý program ještě nějakou další funkci.

Tab. 4 Srovnání emulátorů

Funkce	Net: Netem	WANulator	WANem
Zpoždění	✓	✓	✓
Ztráta paketů	✓	✓	✓
Duplikace paketů	✓	✓	✓
Poškozování Paketů	✓	✓	✓
Změna pořadí paketů	✓	✓	✓
Změna přenosové rychlosti	✗	✓	✓
Filtr na jednotlivé protokoly	✗	✓	✓
Obousměrné změny	✗	✓	✓
Vzdálená Správa	✗	✗	✓
Více síťových karet	✓	✓	✓

4. Závěr

Cílem této bakalářské práce mělo být: Prozkoumat nástroje pro simulaci počítačových sítí a popsat problematiku jejich instalace, provozu a zdokumentovat jejich vlastnosti. Dále potom vytvořit přehled jejich funkcí a vzájemně je porovnat. Při této práci jsem zjistil, že existují dva druhy síťových simulátorů. Jde o klasické simulátory, které podle zadaného skriptu vytvářejí virtuální síťové prostředí, a emulátory, které mění vlastnosti síťových zařízení a pomocí toho simulují různé situace. Z každé kategorie jsem vybral tři zástupce a popsal jejich instalaci a provoz. Dále jsem potom vypracoval srovnání jednotlivých zástupců v dané kategorii a na jednoduchých příkladech předvedl základní funkčnost.

Tato práce může být jednoduchým návodem, jak s těmito programy pracovat. Uživatel se nemusí zdržovat dlouhým studiem manuálů, ale pouze využije mých poznatků k sestavení simulace podle své vlastní potřeby. Nebude se muset rozhodovat, který program zvolit, ale pouze si vybere ten, který bude pro jeho potřeby nejvhodnější. Popisovat všechny vlastnosti jednotlivých programů by bylo nad rámec této práce, ale pokud by uživatel potřeboval sestavovat nějaké složitější skripty, tato práce obsahuje velké množství odkazů na manuály jednotlivých programů, odkud může čerpat další informace.

Literatura

[1] The Network Simulator - ns-2, Oficiální stránky simulátoru Ns-2 (2004) [cit. 2008-10-11] : Dostupné z WWW: <http://www.isi.edu/nsnam/ns/>.

[2] The Manual (formerly ns Notes and Documentation), Dokumentace a manuál Ns-2 (2004) [cit. 2008-11-23] : Dostupné z WWW: <http://www.isi.edu/nsnam/ns/doc/index.html/>.

[3] The ns-3 Network Simulator, Oficiální stránky simulátoru Ns-3 (2004) [cit. 2008-11-5] : Dostupné z WWW: <http://www.nsnam.org>.

[4] ns-3 Documentation, Dokumentace simulátoru Ns-3 (2006) [cit. 2008-11-7] : Dostupné z WWW: < http://www.nsnam.org/documents.html>.

[5] Main page Nsnam, Stránka na wiki ohledně Ns-3 (2006) [cit. 2009-1-5] : Dostupné z WWW: < http://www.nsnam.org/wiki/index.php/Main_Page>.

[6] Mercurial Repositories index, Přehled tříd Ns-3 (2006) [cit. 2009-2-9] : Dostupné z WWW: < http://code.nsnam.org/>.

[7] ns version 1 - LBNL Network Simulator, Oficiální stránky Ns-1 (2001) [cit. 2009-2-19] : Dostupné z WWW: < http://www-nrg.ee.lbl.gov/ns/>.

[8] NS(1) manual page, Dokumentace simulátoru Ns-1 (2001) [cit. 2009-2-20] : Dostupné z WWW: < http://www-nrg.ee.lbl.gov/ns/man.html >.

[9] Net:Netem – The Linux Foundation, Emulátor Net:Netem (2005) [cit. 2009-3-1] : Dostupné z WWW: < http://www.linuxfoundation.org/en/Net:Netem >.

[10] WANulator - Home, Oficiální stránky emulátoru WANulator (2006) [cit. 2009-3-3] : Dostupné z WWW: < http://www.wanulator.de/ >.

[11] WANEM The Wide Area Network Emulator, Oficiální stránky emulátoru WANem (2005) [cit. 2009-3-10] : Dostupné z WWW: < http://wanem.sourceforge.net/ >.

Přílohy

Veškeré přílohy jsou dostupné elektronicky ve formátu PDF na přiloženém disku DVD ve složce přílohy.

[A] Skript pro simulátor Ns-2 – Bezdrátová simulace

[B] Skript pro simulátor Ns-2 – Složitější simulace

[C] Skript pro simulátor Ns-3

[D] Obrázky z instalace WANem

[E] Výsledky provozu WANem

[F] DVD obsahující instalační soubory všech testovaných programů a testovací skripty